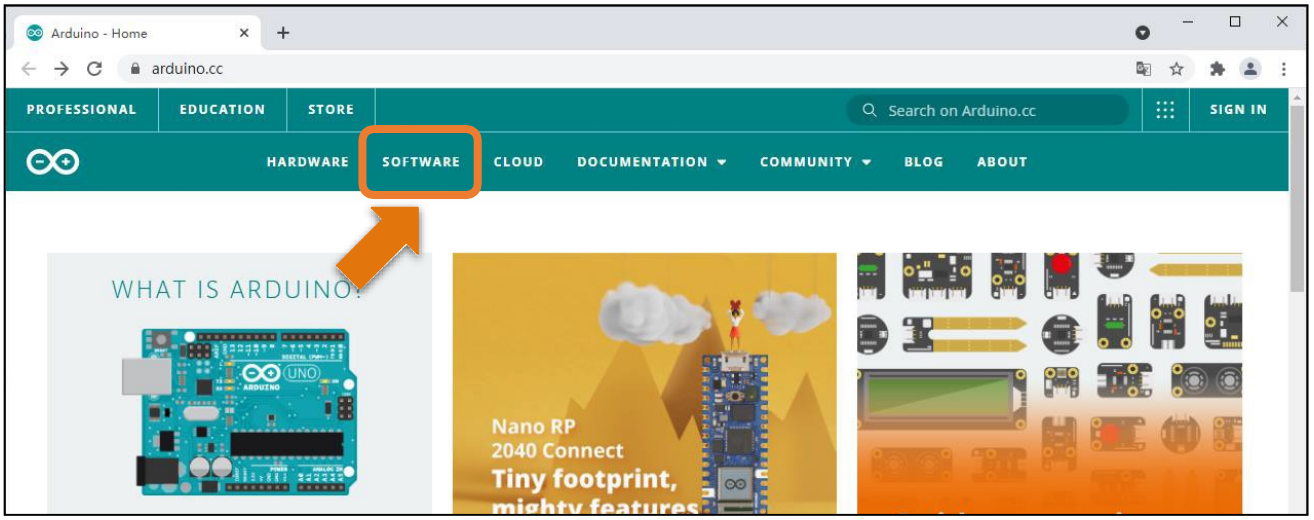
Getting Ready for Arduino

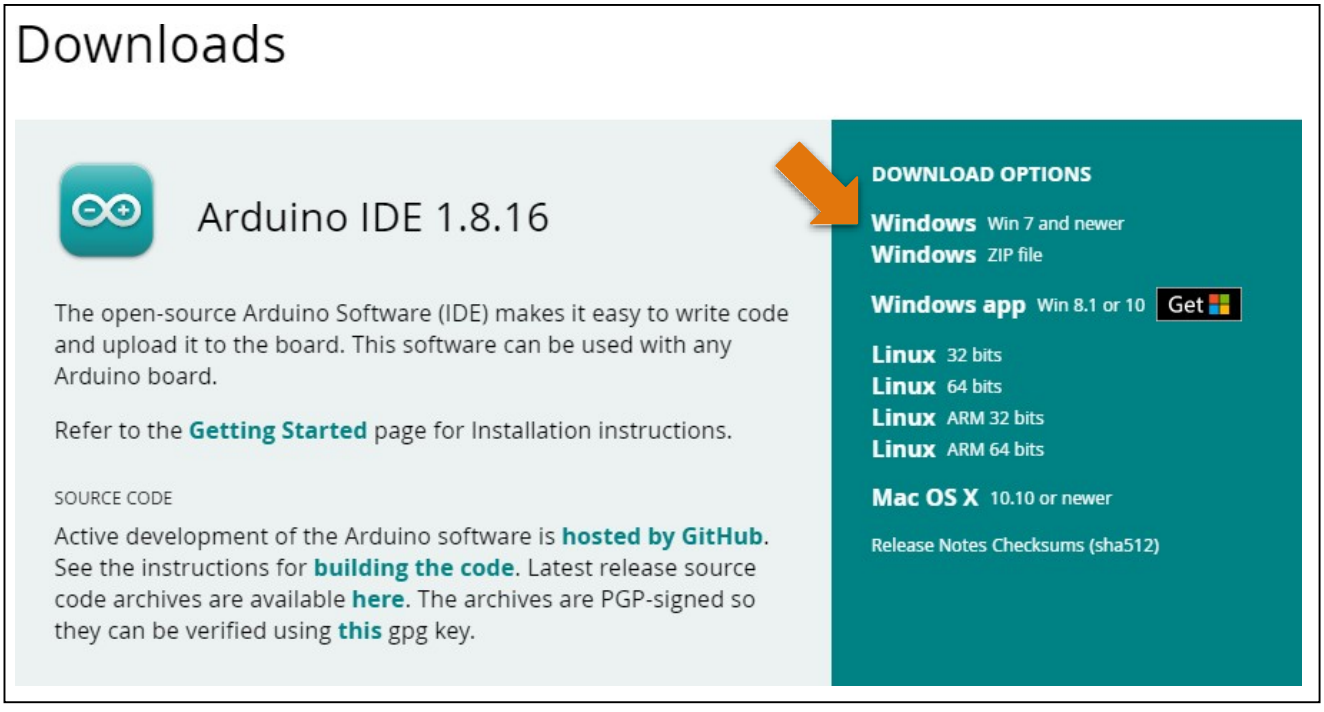
Before starting building the projects, you need to make some preparation first, which is so crucial that you must not skip.

Programming Software

Arduino Software (IDE) is used to write and upload the code for Arduino Board.
First, install Arduino Software (IDE): visit https://www.arduino.cc, click "Download" to enter the download page.

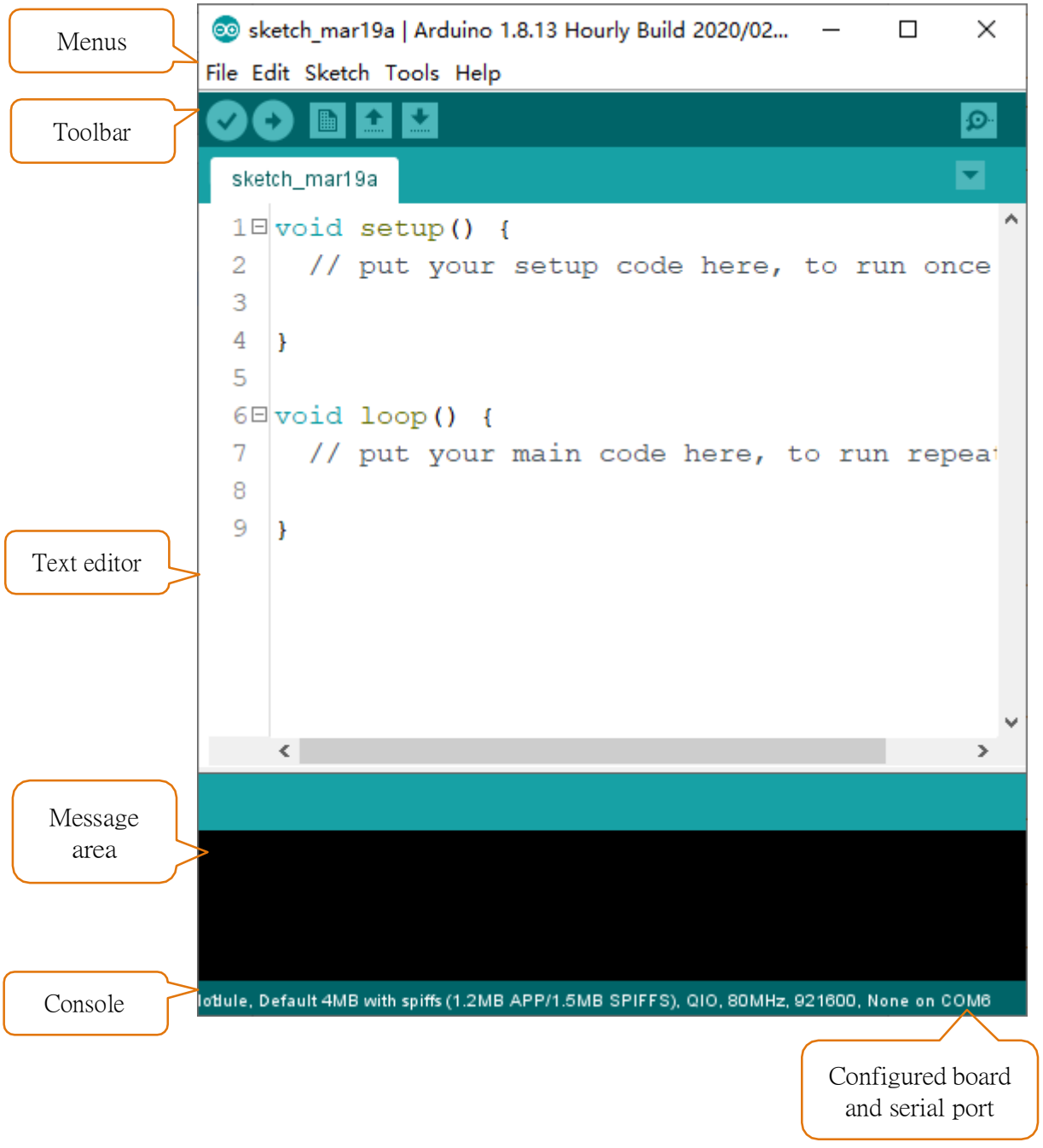


Select and download corresponding installer according to your operating system. If you are a windows user, please select the "Windows Installer" to download to install the driver correctly.



After the download completes, run the installer. For Windows users, there may pop up an installation dialog box of driver during the installation process. When it popes up, please allow the installation.
After installation is complete, an Arduino Software shortcut will be generated in the desktop. Run the Arduino Software.

The interface of Arduino Software is as follows:



Programs written with Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and saved with the file extension **.ino**. The editor has features for cutting/pasting and searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

Verify
Check your code for compile errors .

Upload
Compile your code and upload them to the configured board.

New
Create a new sketch.

Open
Present a menu of all the sketches in your sketchbook. Clicking one will open it within the current window and overwrite its content.

Save
Save your sketch.

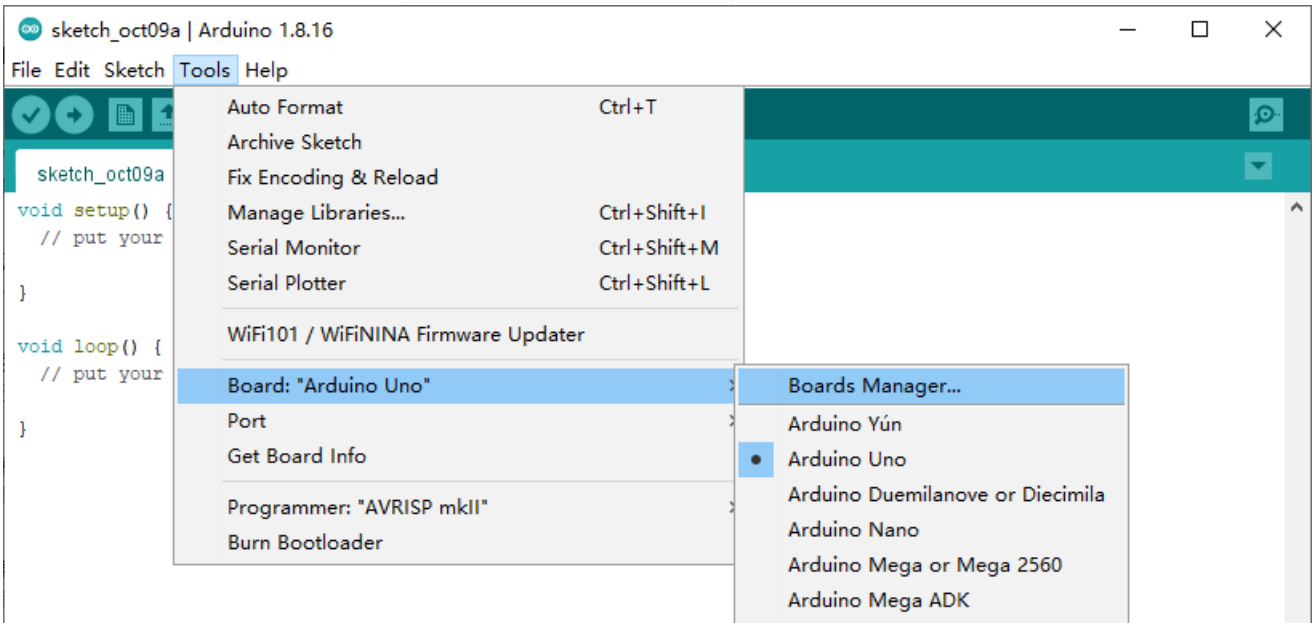
Serial Monitor
Open the serial monitor.

Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

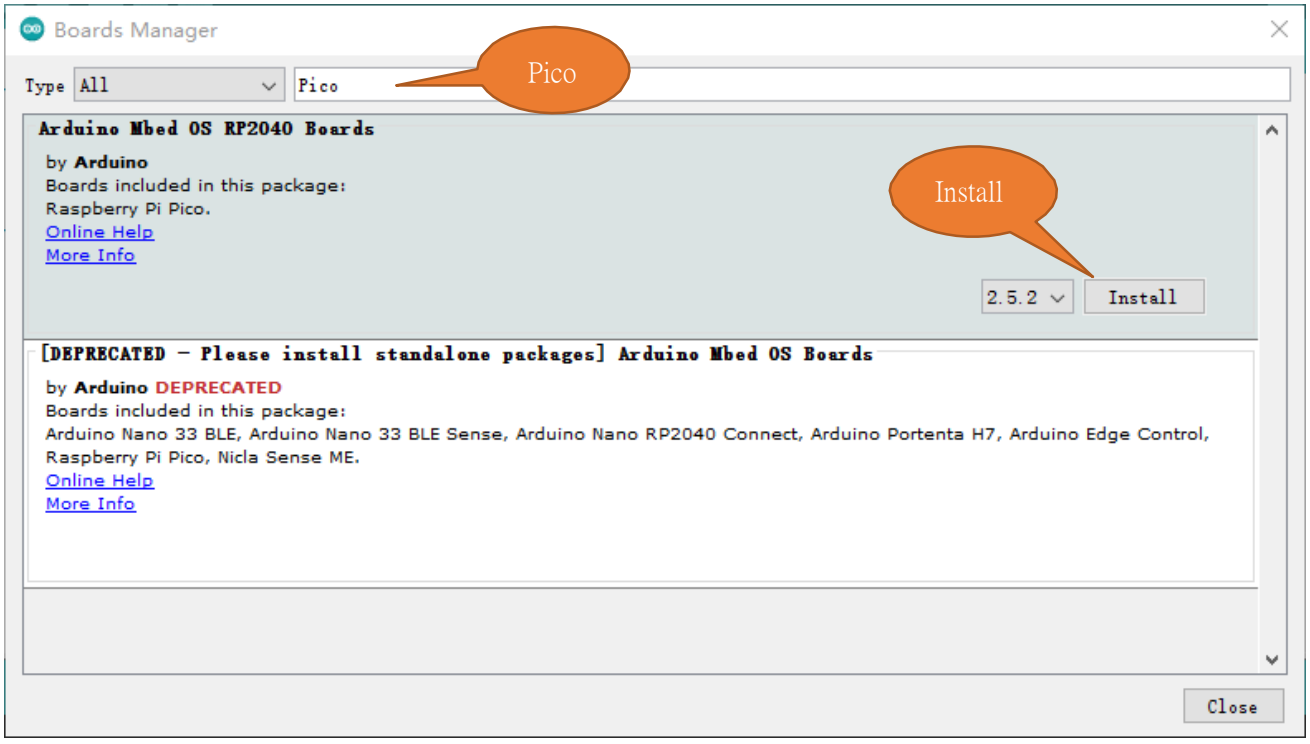
Installation of Development Board Support Package

1, Make sure your network is of good connection.

2, Open Arduino IDE. Click Tools>**Board**>**Boards Manager...**on the menu bar.



3, Enter Pico in the searching box, select "Arduino Mbed OS RP2040 Boards" and click on Install.

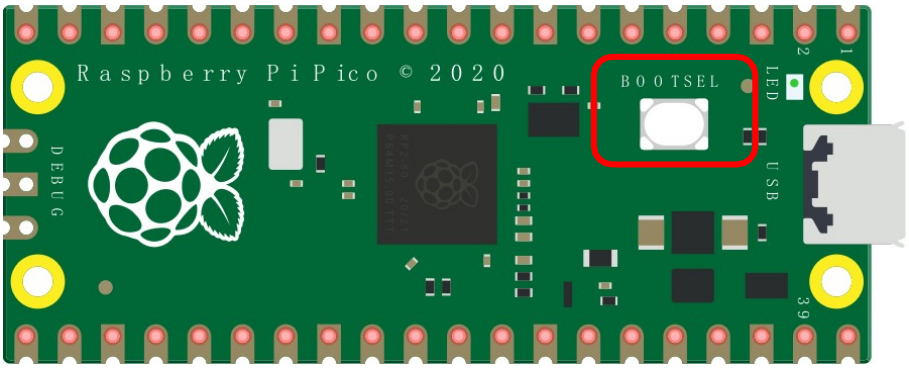


4, Click Yes in the pop-up "**dpinst-amd64.exe**"installation window. (Without it, you will fail to communicate with Arduino.) Thus far, we have finished installing the development support package.

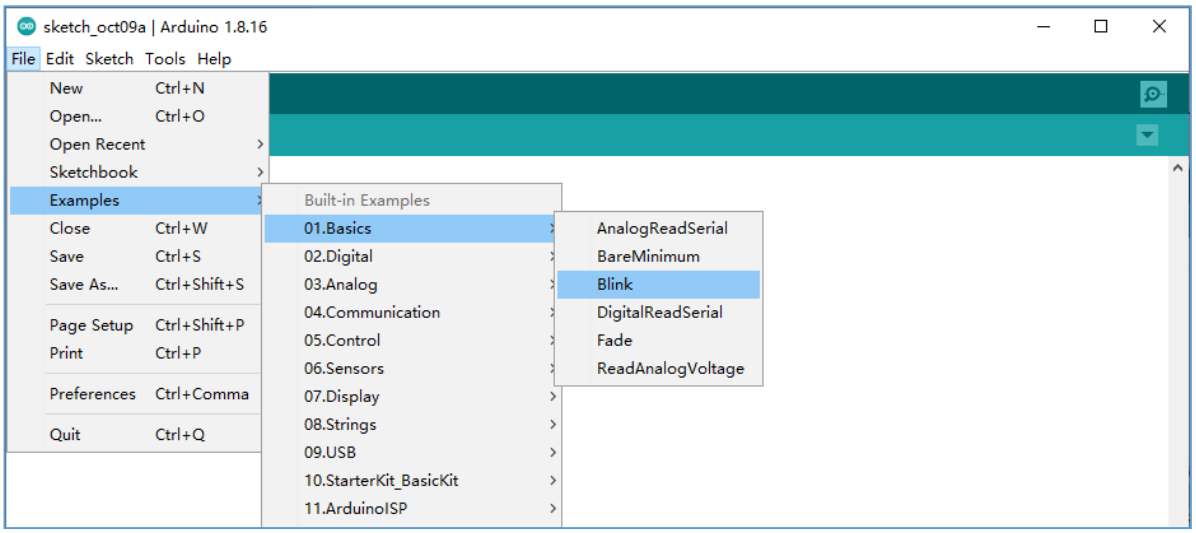
Uploading Adruino-compatible Firmware for Pico

If your Pico is new and you want to use Arduino to learn and develop, you need to upload an Adruino-compatible Firmware for it. Please refer to the following steps to cinfigure.

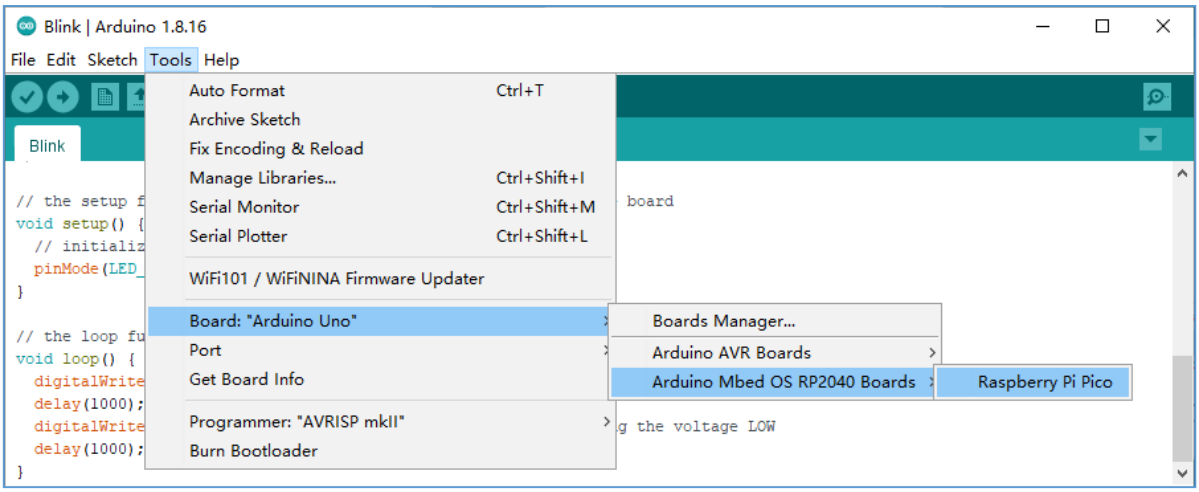
1, Disconnect Pico from computer. Keep pressing the white button(BOOTSEL) on Pico, and connect Pico to computer before releasing the button. (Note: Be sure to keep pressing the button before powering the Pico, otherwise the firmware will not download successfully)



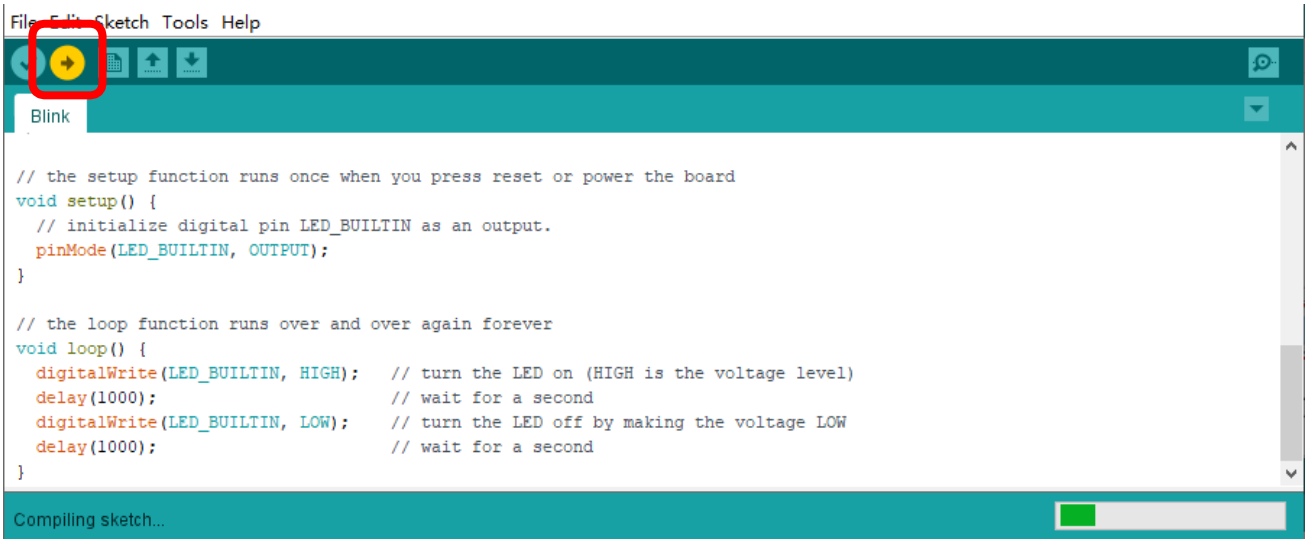
2, Open Arduino IDE. Click File>Examples>01.Basics>Blink.



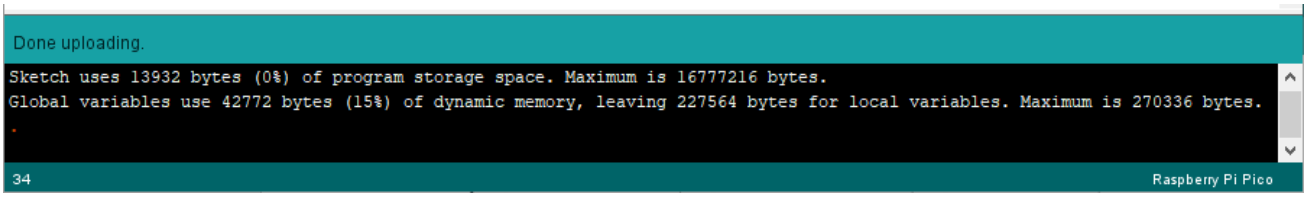
3, Click **Tools>Board>Arduino Mbed OS RP2040 Boards>Raspberry Pi Pico**.



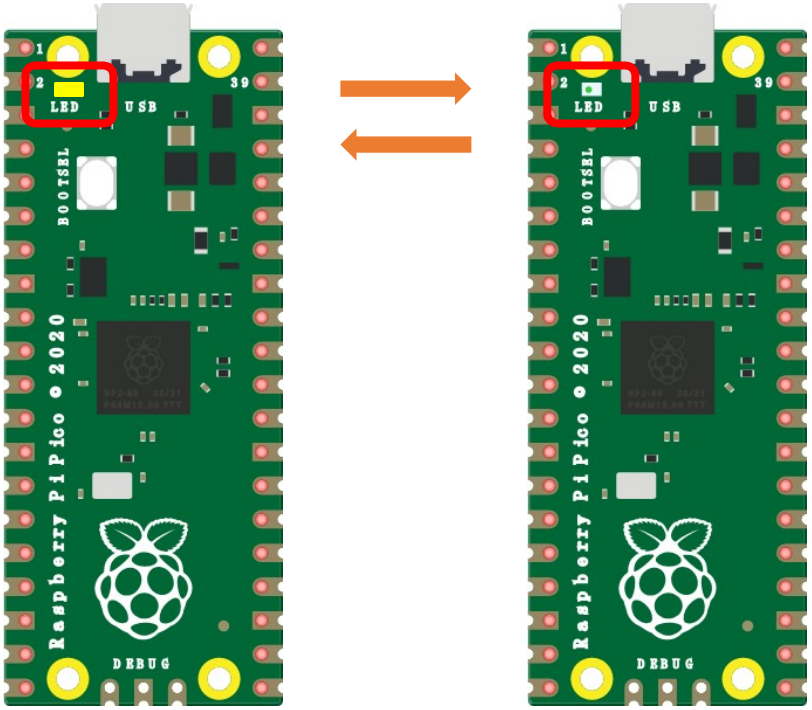
4,Upload sketch to Pico.



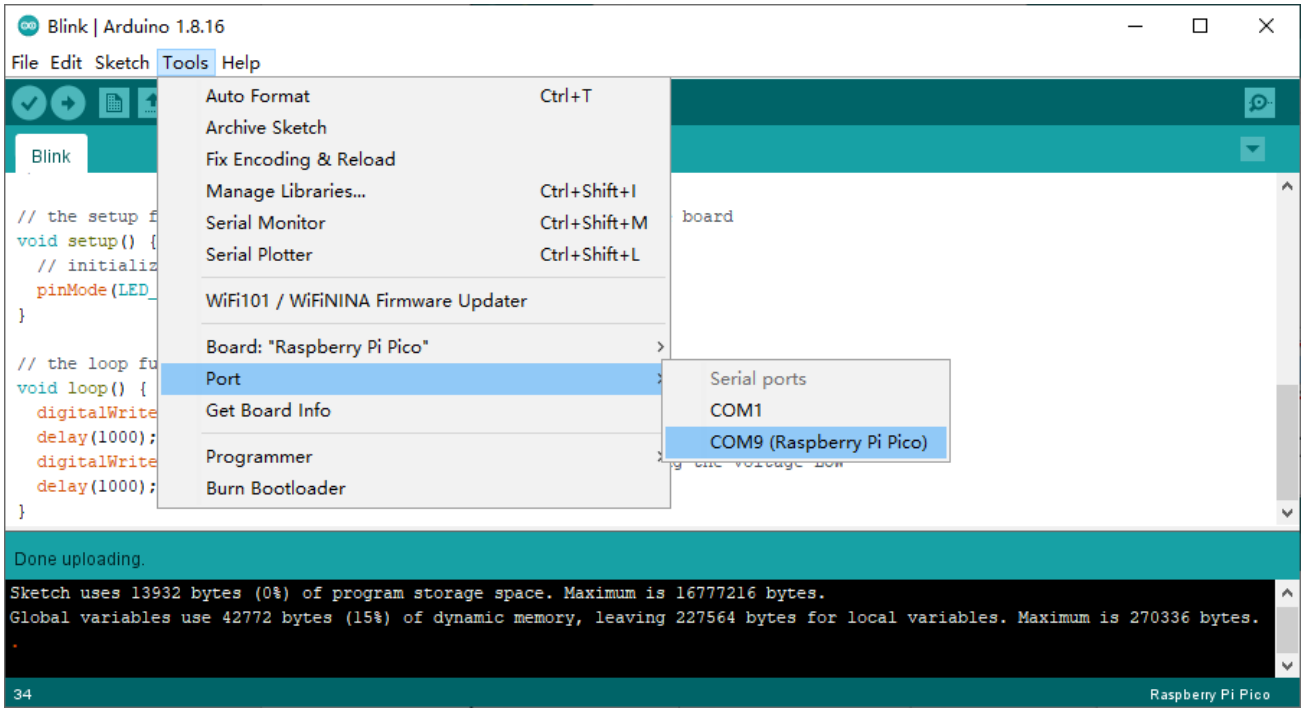
When the sketch finishes uploading, you can see the following prompt.



And the indicator on Pico starts to flash.



5,Click **Tools>Port>COMx(Raspberry Pi Pico)**. X of COMx varies from different computers. Please select the correct one on your computer. In our case, it is COM9.



Note:

1. At the first time you use Arduino to upload sketch for Pico, you don't need to select port. After that, each time before uploading sketch, please check whether the port has beed selected; otherwise, the downloading may fail.
2. Sometimes when using, Pico may lose firmware due to the code and fail to work. At this point, you can upload firmware for Pico as mentioned above

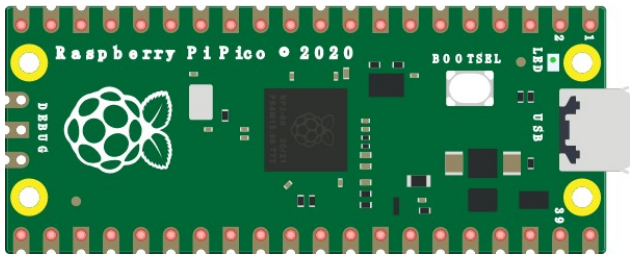
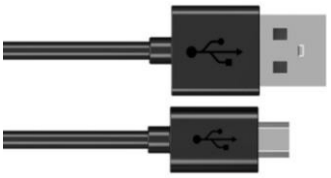
Example

This chapter is the Start Point in the journey to build and explore Pico electronic projects. We will start with simple "Blink" project.

Project 1.1 Blink

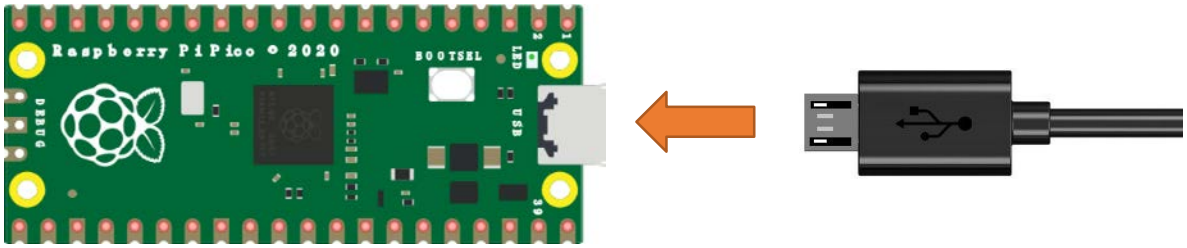
In this project, we will use Raspberry Pi Pico to control blinking a common LED.

Component List

Raspberry Pi Picox1 	USB cable x1 

Power

Raspberry Pi Pico requires 5V power supply. You can either connect external 5V power supply to Vsys pin of Pico or connect a USB cable to the onboard USB base to power Pico.
In this tutorial, we use USB cable to power Pico and upload sketches.



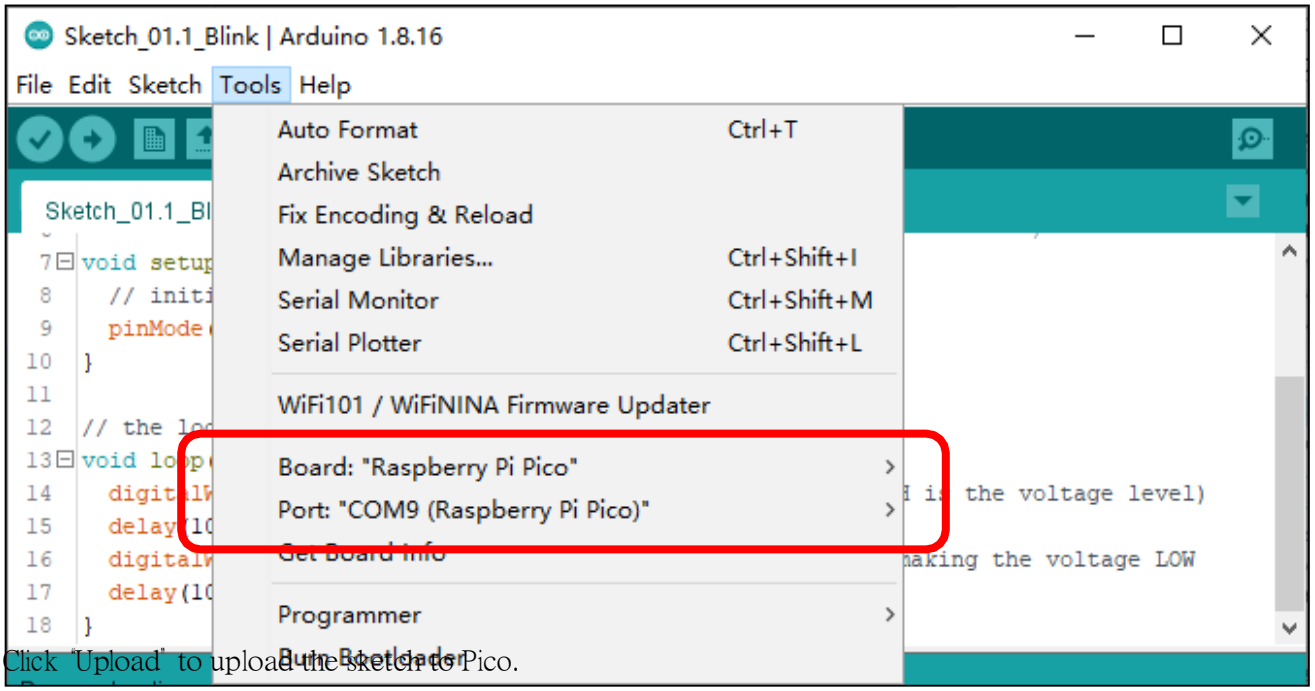
Sketch

The onboard LED of Raspberry Pi Pico is controlled by GP25. When GP25 outputs high level, LED lights up; When it outputs low, LED lights off. You can open the provided code:

Raspberry_Pi _Pico Kit Tutorial\Lesson1-C\Sketches\Sketch_01.1_Blink

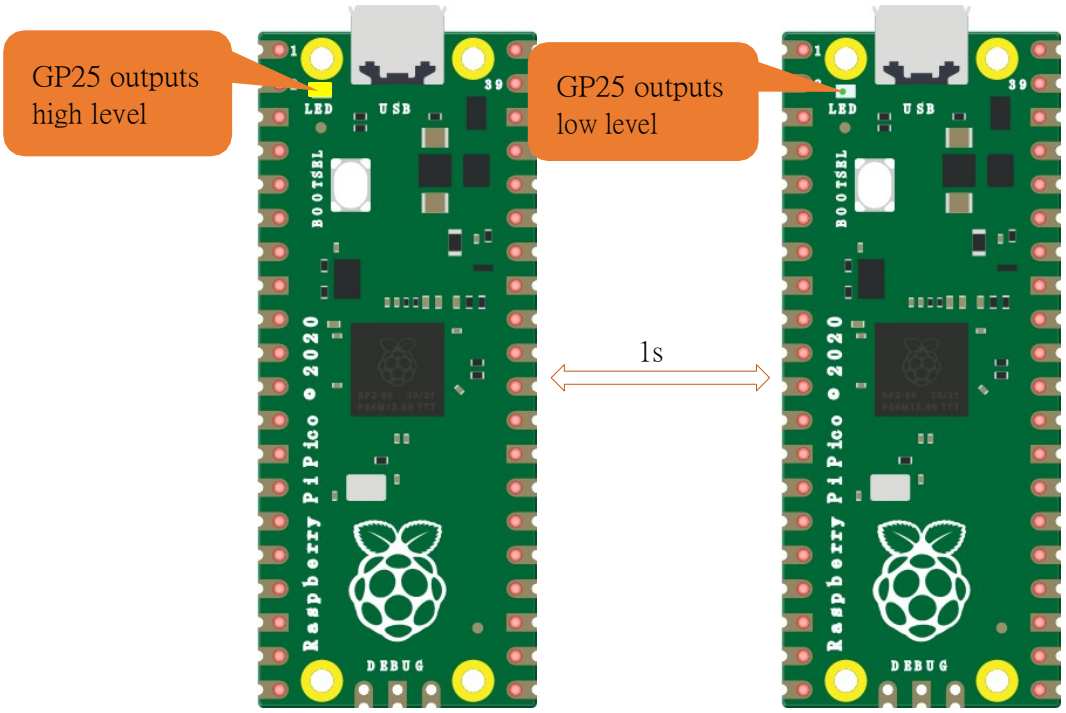
Before uploading code to Pico, please check the configuration of Arduino IDE.

Click Tools, make sure Board and Port are as follows:



Click "Upload" to upload the sketch to Pico.

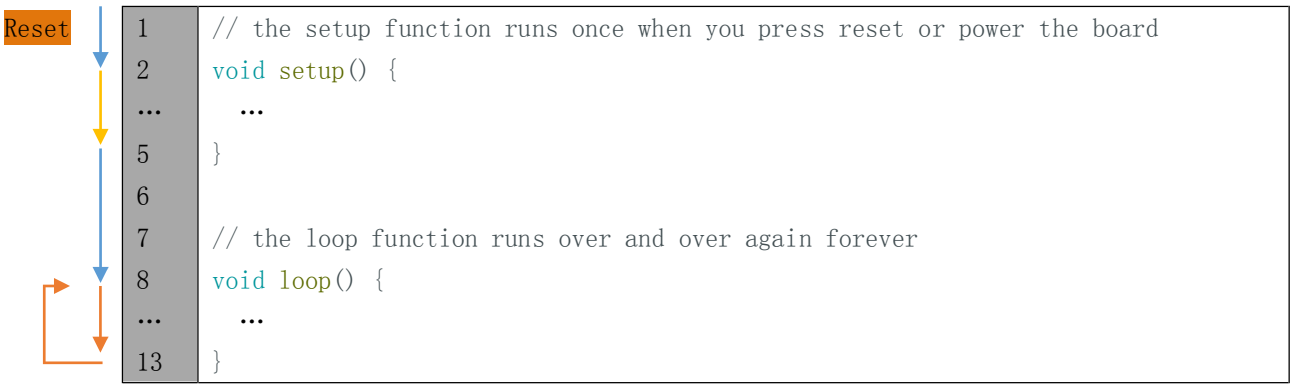
Pico's on-board LED lights on and off every 1s, flashing cyclically.



The following is the program code:

```
1   #define LED_BUILTIN 25
2
3   // the setup function runs once when you press reset or power the board
4   void setup() {
5     // initialize digital pin LED_BUILTIN as an output.
6     pinMode(LED_BUILTIN, OUTPUT);
7   }
8
9   // the loop function runs over and over again forever
10  void loop() {
11    digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
12    delay(1000); // wait for a second
13    digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
14    delay(1000); // wait for a second
15  }
```

The Arduino IDE code usually contains two basic functions: void setup() and void loop().
After the board is reset, the setup() function will be executed firstly, and then the loop() function.
setup() function is generally used to write code to initialize the hardware. And loop() function is used to write code to achieve certain functions. loop() function is executed repeatedly. When the execution reaches the end of loop(), it will back to the beginning of loop() to run again.



In the circuit, GP25 of Pico is connected to the LED, so the LED pin is defined as 25.

```
1   #define LED_BUILTIN 25
```

This means that after this line of code, all LED_BUILTIN will be regarded as 25.
In the setup() function, first, we set the LED_BUILTIN as output mode, which can make the port output high or low level.

```
4     // initialize digital pin LED_BUILTIN as an output.
5     pinMode(LED_BUILTIN, OUTPUT);
```

Then, in the loop() function, set the LED_BUILTIN to output high level to make LED light up.

```
10    digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
```

Wait for 1000ms, that is 1s. Delay() function is used to make control board wait for a moment before executing the next statement. The parameter indicates the number of milliseconds to wait for.

```
11    delay(1000); // wait for a second
```

Then set the LED_BUILTIN to output low level, and LED lights off. One second later, the execution of loop() function will be completed.

```
12    digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
13    delay(1000); // wait for a second
```

The loop() function is constantly being executed, so LED will keep blinking.

Reference

void pinMode(int pin, int mode);

Configures the specified pin to behave either as an input or an output.

Parameters

pin: the pin number to set the mode of LED.

mode: INPUT, OUTPUT, INPUT_PULLDOWM, or INPUT_PULLUP.

void digitalWrite (int pin, int value);

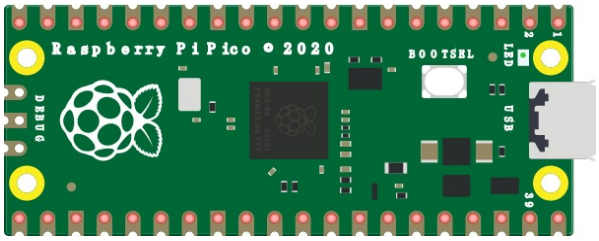
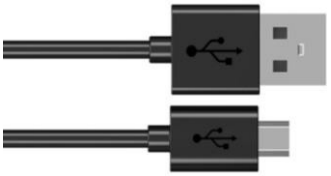
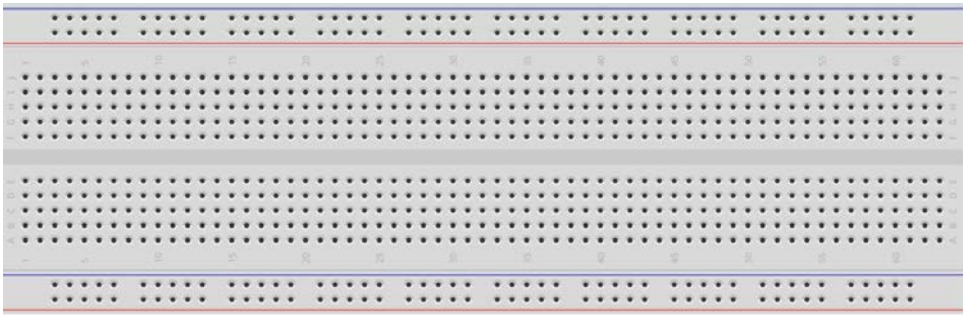
Writes the value HIGH or LOW (1 or 0) to the given pin which must have been previously set as an output.

For more related functions, please refer to https://www.arduino.cc/reference/en/

Project 1.2 Blink

In this project, we will use Raspberry Pi Pico to control blinking a common LED.

Component List

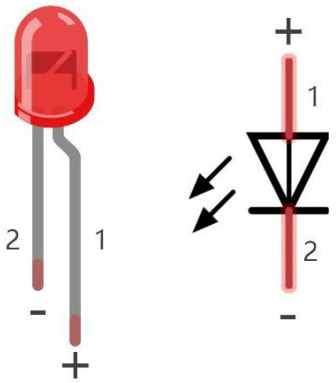
Raspberry Pi Pico x1 	USB Cable x1 	
Breadboard x1 		
LED x1 	Resistor 220Ω x1 	Jumper 

Component Knowledge

LED

An LED is a type of diode. All diodes only work if current is flowing in the correct direction and have two Poles. An LED will only work (light up) if the longer pin (+) of LED is connected to the positive output from a power source and the shorter pin is connected to the negative (-). Negative output is also referred to as Ground (GND). This type of component is known as "Polar" (think One-Way Street).

All common 2 lead diodes are the same in this respect. Diodes work only if the voltage of its positive electrode is higher than its negative electrode and there is a narrow range of operating voltage for most all common diodes of 1.9 and 3.4V. If you use much more than 3.3V the LED will be damaged and burn out.



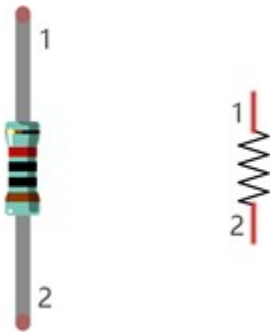
LED	Voltage	Maximum current	Recommended current
Red	1.9-2.2V	20mA	10mA
Green	2.9-3.4V	10mA	5mA
Blue	2.9-3.4V	10mA	5mA
Volt ampere characteristics conform to diode			

Note: LEDs cannot be directly connected to a power supply, which usually ends in a damaged component. A resistor with a specified resistance value must be connected in series to the LED you plan to use.

Resistor

Resistors use Ohms (Ω) as the unit of measurement of their resistance (R). 1MΩ=1000kΩ, 1kΩ=1000Ω.

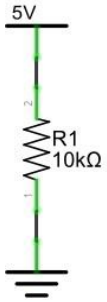
A resistor is a passive electrical component that limits or regulates the flow of current in an electronic circuit. On the left, we see a physical representation of a resistor, and the right is the symbol used to represent the presence of a resistor in a circuit diagram or schematic.



The bands of color on a resistor is a shorthand code used to identify its resistance value. For more details of resistor color codes, please refer to the appendix of this tutorial.

With a fixed voltage, there will be less current output with greater resistance added to the circuit. The relationship between Current, Voltage and Resistance can be expressed by this formula: I=V/R known as Ohm's Law where I = Current, V = Voltage and R = Resistance. Knowing the values of any two of these allows you to solve the value of the third.

In the following diagram, the current through R1 is: I=U/R=5V/10kΩ=0.0005A=0.5mA.

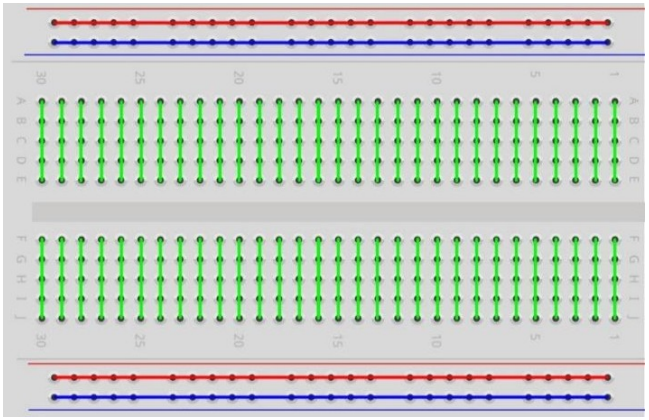


WARNING: Never connect the two poles of a power supply with anything of low resistance value (i.e. a metal object or bare wire) this is a Short and results in high current that may damage the power supply and electronic components.

Note: Unlike LEDs and Diodes, Resistors have no poles and re non-polar (it does not matter which direction you insert them into a circuit, it will work the same)

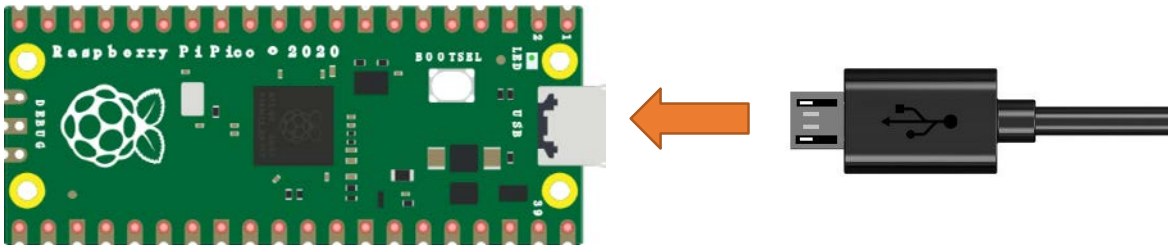
Breadboard

Here we have a small breadboard as an example of how the rows of holes (sockets) are electrically attached. The left picture shows the way to connect pins. The right picture shows the practical internal structure.



Power

In this tutorial, we connect Raspberry Pi Pico and computer with a USB cable.

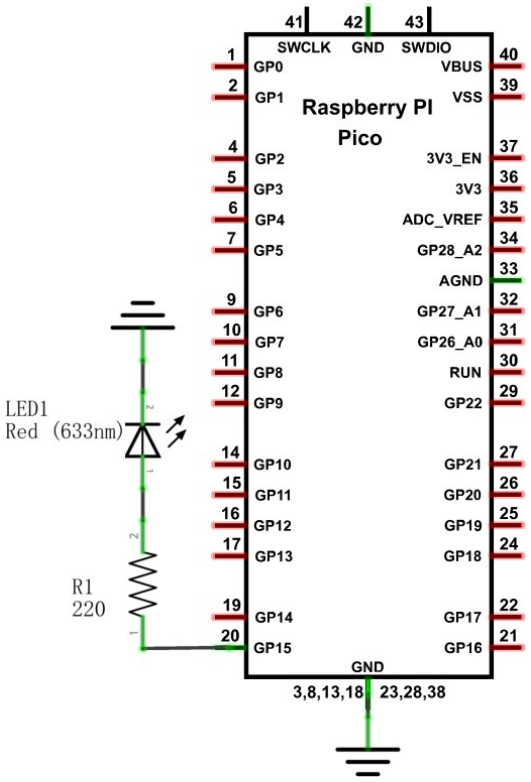


Circuit

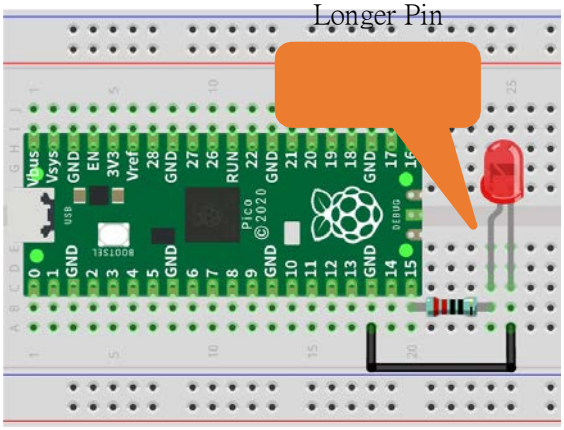
First, disconnect all power from the Raspberry Pi Pico. Then build the circuit according to the circuit and hardware diagrams. After the circuit is built and verified correct, connect the PC to Raspberry Pi Pico.

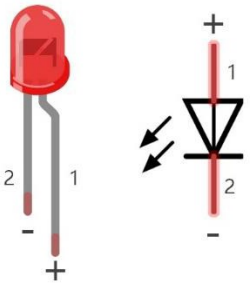
CAUTION: Avoid any possible short circuits (especially connecting 3.3V and GND)!
WARNING: A short circuit can cause high current in your circuit, create excessive component heat and cause permanent damage to your hardware!

Schematic diagram



Hardware connection.





Note: To help users have a better experience when doing the projects, we have made some modifications to Pico's simulation diagram. Please note that there are certain differences between the simulation diagram and the actual board to avoid misunderstanding.

Sketch

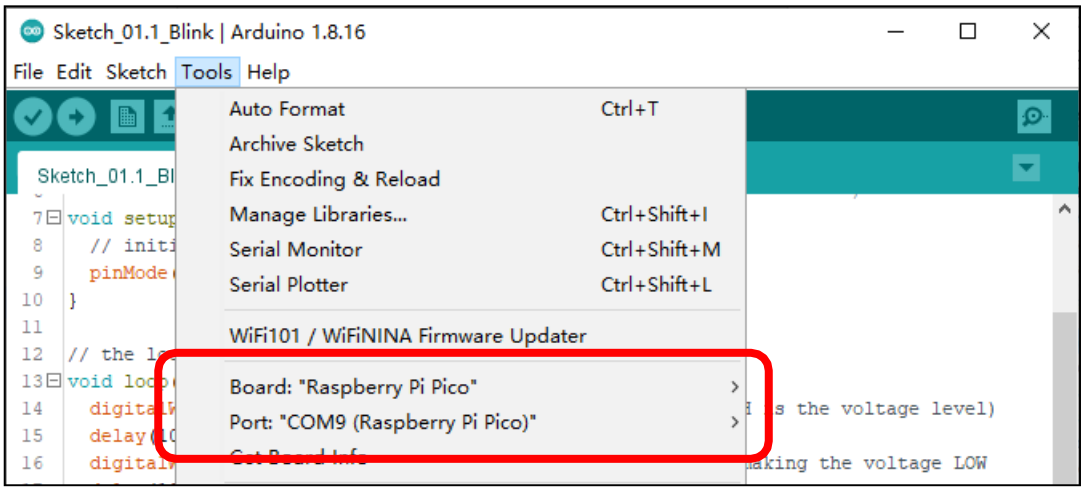
According to the circuit diagram, when GP15 of Pico outputs high level, LED lights up; when it outputs low, LED lights off. Therefore, we can make LED flash repeatedly by controlling GP15 to output high and low repeatedly.

You can open the provided code:

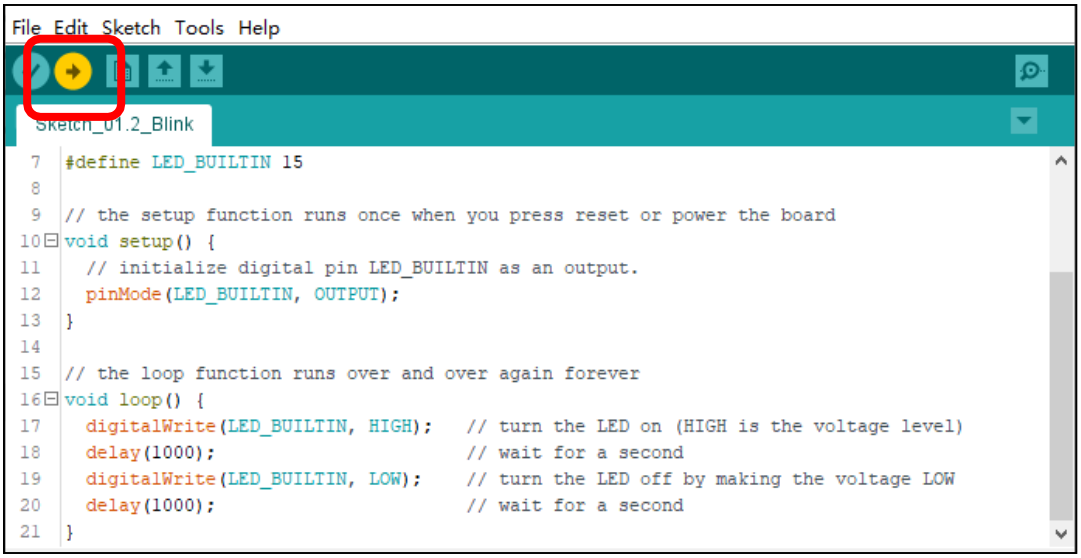
Raspberry_Pi _Pico Kit Tutorial\Lesson1-C\Sketches\Sketch_01.2_Blink

Before uploading code to Pico, please check the configuration of Arduino IDE.

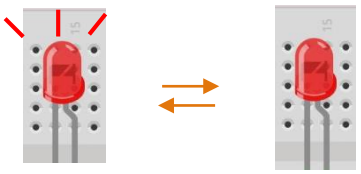
Click Tools, make sure Board and Port are as follows:



Click "Upload" to upload the sketch to Pico.



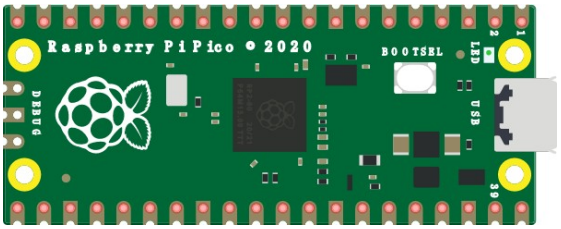
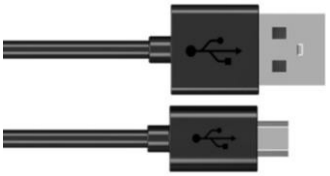
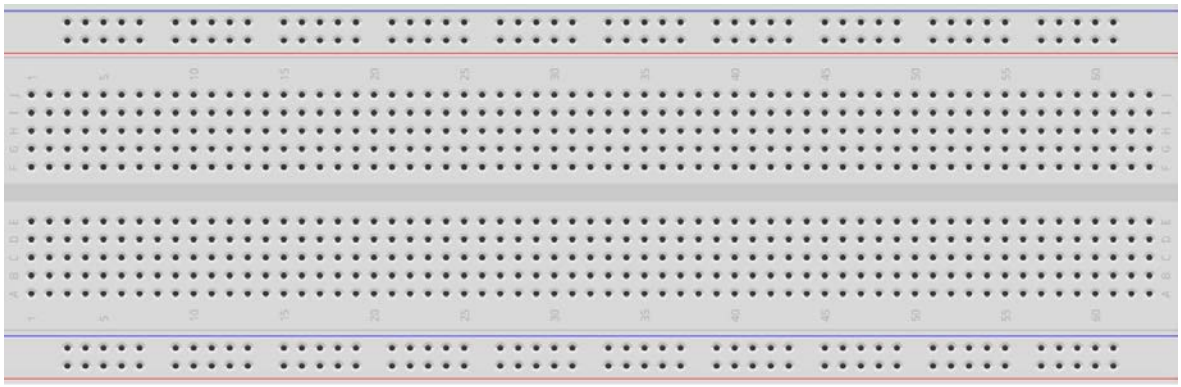
Click "Upload". Download the code to Pico and your LED in the circuit starts Blink.



Project 2.1 Button & LED

In the project, we will control the LED state through a Push Button Switch. When the button is pressed, our LED will turn ON, and when it is released, the LED will turn OFF.

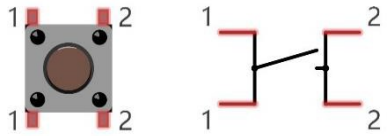
Component List

Raspberry Pi Pico x1 				USB cable x1 
Breadboard x1 				
Jumper 	LED x1 	Resistor 220Ω x1 	Resistor 10kΩ x2 	Push button x1 

Component Knowledge

Push button

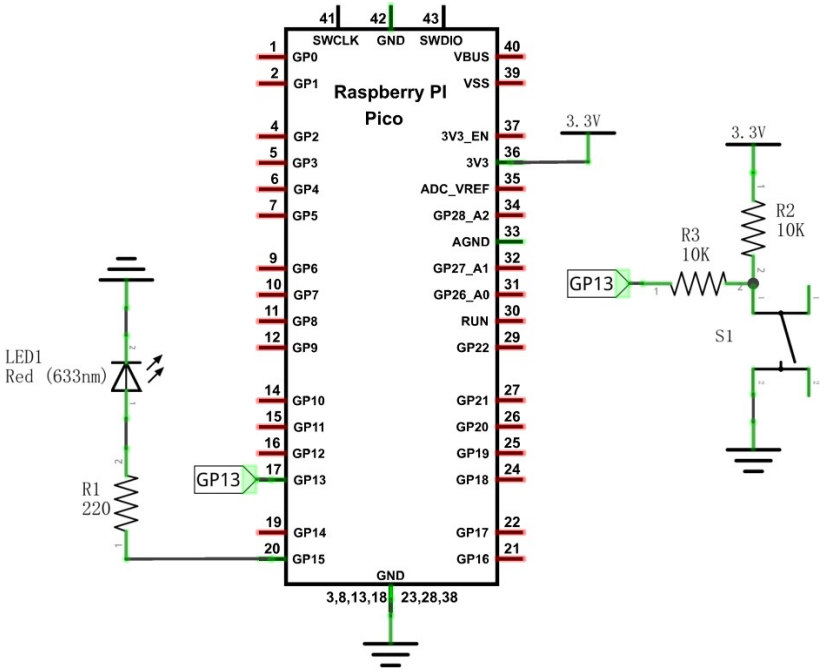
This type of Push Button Switch has 4 pins (2 Pole Switch). Two pins on the left are connected, and both left and right sides are the same per the illustration:



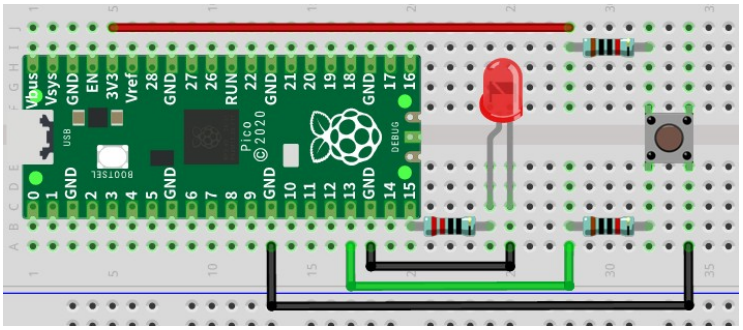
When the button on the switch is pressed, the circuit is completed (your project is Powered ON).

Circuit

Schematic diagram



Hardware connection.



Note: To help users have a better experience when doing the projects, we have made some modifications to Pico's simulation diagram. Please note that there are certain differences between the simulation diagram and the actual board to avoid misunderstanding.

Sketch

This project is designed for learning how to use push button switch to control an LED. We first need to read the state of switch, and then determine whether to turn the LED ON in accordance to the state of the switch.
Upload following sketch:

Raspberry_Pi _Pico Kit Tutorial\Lesson1-C\Sketches\Sketch_02.1_ButtonAndLed

Sketch_02.1_ButtonAndLed | Arduino 1.8.12

File Edit Sketch Tools Help

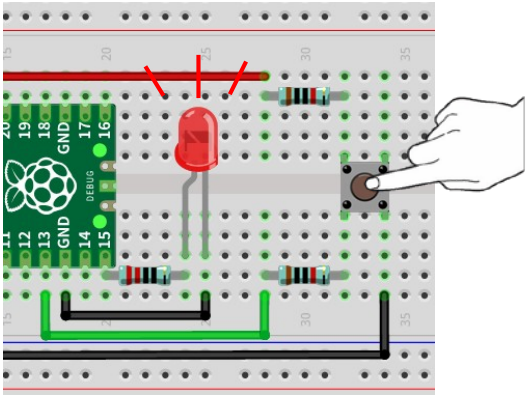
Sketch_02.1_ButtonAndLed

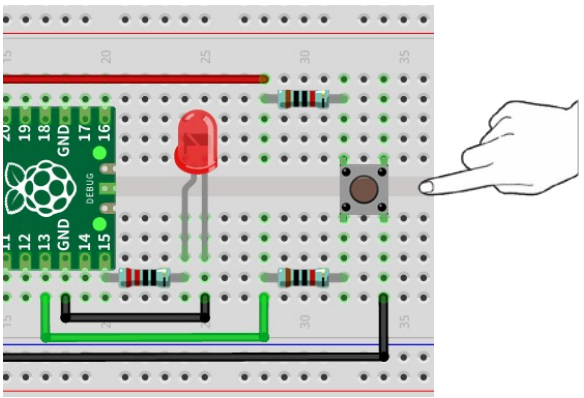
```
#define PIN_LED    15
#define PIN_BUTTON 13

void setup() {
  // initialize digital pin PIN_LED as an output.
  pinMode(PIN_LED, OUTPUT);
  pinMode(PIN_BUTTON, INPUT);
}

// the loop function runs over and over again forever
void loop() {
  if (digitalRead(PIN_BUTTON) == LOW) {
    digitalWrite(PIN_LED,HIGH);
  }else{
    digitalWrite(PIN_LED,LOW);
  }
}
```

Upload the sketch to Pico. When pressing the button, LED lights up; when releasing the button, LED lights OFF.





The following is the program code:

```
#define PIN_LED  15
#define PIN_BUTTON 13
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin PIN_LED as an output.
  pinMode(PIN_LED, OUTPUT);
  pinMode(PIN_BUTTON, INPUT);
}

// the loop function runs over and over again forever
void loop() {
  if (digitalRead(PIN_BUTTON) == LOW) {
    digitalWrite(PIN_LED,HIGH);
  }else{
    digitalWrite(PIN_LED,LOW);
  }
}
```

In the circuit connection, LED and button are connected with GP15 and GP13 respectively, so define ledPin and buttonPin as 15 and 13 respectively.

```
#define PIN_LED  15
#define PIN_BUTTON 13
```

In the while cycle of main function, use digitalRead(buttonPin) to determine the state of button. When the button is pressed, the function returns low level and the result of "if" is true, so LED lights up. Otherwise, LED lights OFF.

```
void loop() {
  if (digitalRead(PIN_BUTTON) == LOW) {
    digitalWrite(PIN_LED,HIGH);
  }else{
    digitalWrite(PIN_LED,LOW);
  }
}
```

Reference

int digitalRead (int pin);

This function returns the value read at the given pin. It will be "HIGH" or "LOW" (1 or 0) depending on the logic level at the pin.